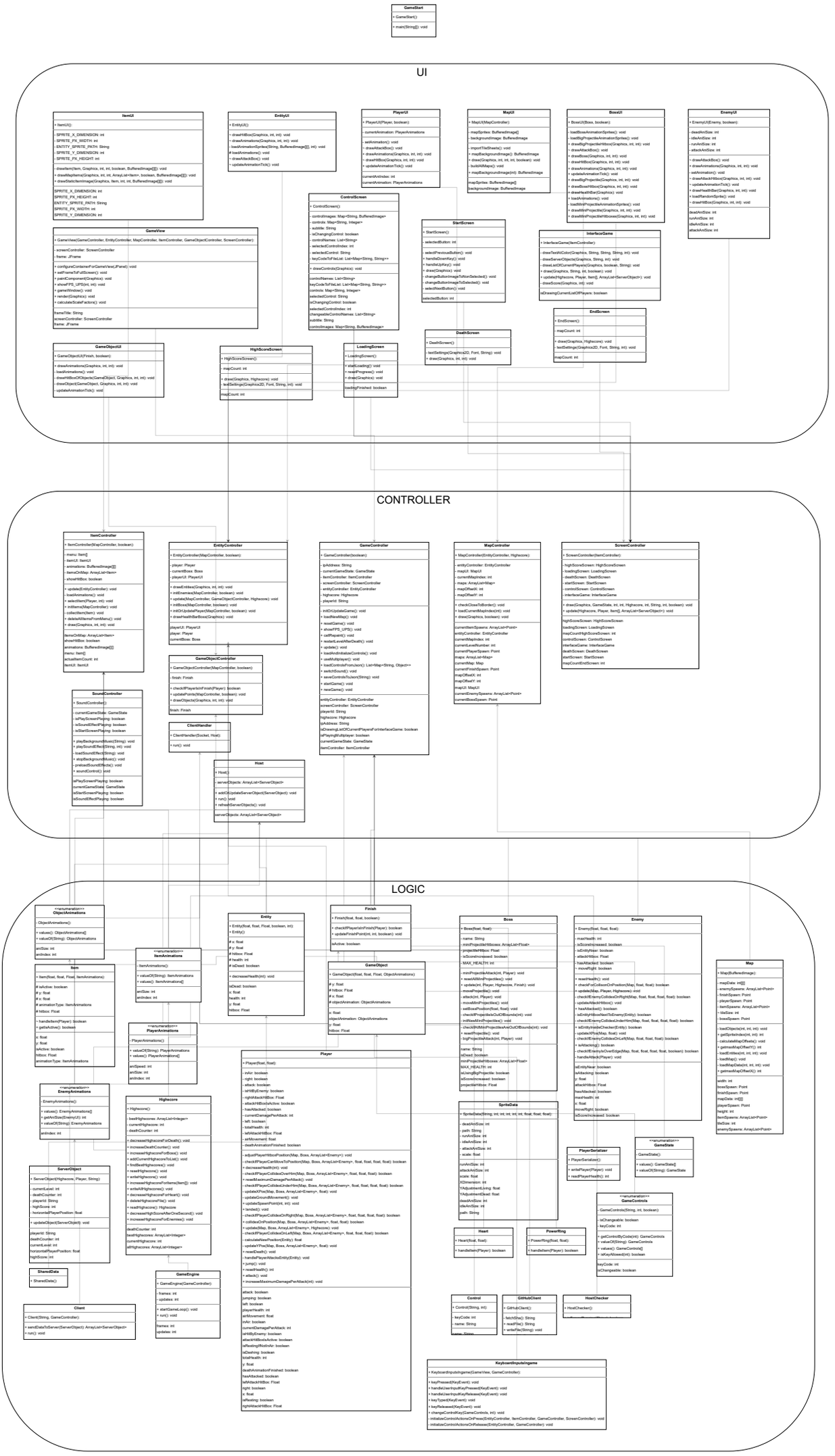
GameStart
+ GameStart()
+ main(String[]): void
UI
ItemUI
+ ItemUI()
- SPRITE_X_DIMENSION: int
- SPRITE_PX_HEIGHT: int
- ENTITY_SPRITE_PATH: String
- SPRITE_Y_DIMENSION: int
- SPRITE_PX_HEIGHT: int
- drawItem(Item, Graphics, int, int, boolean, BufferedImage[]): void
+ drawAspitems(Graphics, int, int, ArrayList<Item>, boolean, BufferedImage[]): void
+ drawStaticItemImage(Graphics, Item, int, int, BufferedImage[]): void
SPRITE_X_DIMENSION: int
SPRITE_PX_HEIGHT: int
ENTITY_SPRITE_PATH: String
SPRITE_PX_WIDTH: int
SPRITE_Y_DIMENSION: int
EntityUI
+ EntityUI()
~ drawHitBox(Graphics, int, int): void
~ drawAnimations(Graphics, int, int): void
- loadAnimationSprites(String, BufferedImage[][], int): void
# loadAnimations(): void
- drawAttackBox(): void
- updateAnimationTick(): void
PlayerUI
+ PlayerUI(Player, boolean):
- currentAnimation: PlayerAnimations
- setAnimation(): void
+ drawAttackBox(): void
+ drawAnimations(Graphics, int, int): void
+ drawHitBox(Graphics, int, int): void
+ updateAnimationTick(): void
currentAniIndex: int
currentAnimation: PlayerAnimations
MapUI
+ MapUI(MapController):
- mapSprites: BufferedImage[]
- backgroundImage: BufferedImage
- importTileSheets(): void
+ mapBackgroundImage(): BufferedImage
- draw(Graphics, int, int, int, boolean): void
- buildAllMaps(): void
+ mapBackgroundImage(int): BufferedImage
mapSprites: BufferedImage[]
backgroundImage: BufferedImage
BossUI
+ BossUI(Boss, boolean):
- loadBossAnimationSprites(): void
- loadBigProjectileAnimationSprites(): void
+ drawBigProjectileHitbox(Graphics, int, int): void
+ drawAttackBox(): void
+ drawBoss(Graphics, int, int): void
+ drawHitBox(Graphics, int, int): void
+ drawAnimations(Graphics, int, int): void
+ updateAnimationTick(): void
+ drawBigProjectile(Graphics, int, int): void
+ drawBossHitbox(Graphics, int, int): void
+ drawHealthBar(Graphics): void
+ loadAnimations(): void
- loadMinProjectileAnimationSprites(): void
+ drawMinProjectile(Graphics, int, int): void
+ drawMinProjectileHitboxes(Graphics, int, int): void
EnemyUI
+ EnemyUI(Enemy, boolean):
- deadAniSize: int
- idleAniSize: int
- runAniSize: int
- attackAniSize: int
- drawAttackBox(): void
+ drawAnimations(Graphics, int, int): void
- setAnimation(): void
- drawAttackHitbox(Graphics, int, int): void
- updateAnimationTick(): void
+ drawHealthBar(Graphics, int, int): void
+ loadRandomSprites(): void
+ drawHitBox(Graphics, int, int): void
deadAniSize: int
runAniSize: int
idleAniSize: int
attackAniSize: int
ControlScreen
+ ControlScreen():
- controlImages: Map<String, BufferedImage>
- controls: Map<String, Integer>
- subtitle: String
- isChangingControl: boolean
- controlNames: List<String>
- selectedControlIndex: int
- selectedControl: String
- keyCodeToFileList: List<Map<String, String>>
+ drawControls(Graphics): void
controlNames: List<String>
keyCodeToFileList: List<Map<String, String>>
controls: Map<String, Integer>
selectedControl: String
isChangingControl: boolean
selectedControlIndex: int
changeableControlNames: List<String>
subtitle: String
controlImages: Map<String, BufferedImage>
StartScreen
+ StartScreen():
- selectedButton: int
- selectPreviousButton(): void
+ handleDownKey(): void
+ handleUpKey(): void
+ draw(Graphics): void
- changeButtonImageToNonSelected(): void
- changeButtonImageToSelected(): void
- selectNextButton(): void
selectedButton: int
InterfaceGame
+ InterfaceGame(ItemController):
- drawTextAtColor(Graphics, String, String, String, int): void
- drawServerObjects(Graphics, String, int): void
- drawLeftOfCurrentPlayers(Graphics, boolean, String): void
+ draw(Graphics, String, int, boolean): void
+ update(Highscore, Player, Item[], ArrayList<ServerObject>): void
- drawScore(Graphics, int): void
isDrawingCurrentListOfPlayers: boolean
GameView
+ GameView(GameController, EntityController, MapController, ItemController, GameObjectController, ScreenController):
- screenController: ScreenController
- frame: JFrame
+ configureContainerForGameView(JFrame): void
+ setFrameToFullScreen(): void
+ paintComponent(Graphics): void
+ showFPS_UPS(int, int): void
+ gameWindow(): void
+ render(Graphics): void
+ calculateScaleFactors(): void
itemsTitle: String
screenController: ScreenController
frame: JFrame
EndScreen
+ EndScreen():
- mapCount: int
+ draw(Graphics, Highscore): void
- textSettings(Graphics2D, Font, String, int): void
mapCount: int
DeathScreen
+ DeathScreen():
- textSettings(Graphics2D, Font, String): void
+ draw(Graphics, int, int): void
GameObjectUI
+ GameObjectUI(Finish, boolean):
+ drawAnimations(Graphics, int, int): void
- loadAnimations(): void
- drawHitBoxOfObjects(GameObject, Graphics, int, int): void
- drawObject(GameObject, Graphics, int, int): void
- updateAnimationTick(): void
HighScoreScreen
+ HighScoreScreen():
- mapCount: int
+ draw(Graphics, Highscore): void
+ textSettings(Graphics2D, Font, String, int): void
mapCount: int
LoadingScreen
+ LoadingScreen():
+ startLoading(): void
+ nextProgress(): void
+ draw(Graphics): void
loadingFinished: boolean
CONTROLLER
ItemController
+ ItemController(MapController, boolean):
- menu: Item[]
- itemUI: ItemUI
- animations: BufferedImage[][]
- itemsOnMap: ArrayList<Item>
- showHitBox: boolean
+ update(EntityController): void
- loadAnimations(): void
+ selectItem(Player, int): void
+ initItems(MapController): void
- collectItem(Item): void
+ deleteAllItemsFromMenu(): void
+ draw(Graphics, int, int): void
itemsOnMap: ArrayList<Item>
showHitBox: boolean
animations: BufferedImage[][]
menu: Item[]
actualItemCount: int
itemUI: ItemUI
EntityController
+ EntityController(MapController, boolean):
- player: Player
- currentBoss: Boss
- playerUI: PlayerUI
+ drawEntities(Graphics, int, int): void
+ initEnemies(MapController, boolean): void
+ update(MapController, GameObjectController, Highscore): void
+ initBoss(MapController, boolean): void
+ initOrUpdatePlayer(MapController, boolean): void
+ drawHealthBarBoss(Graphics): void
playerUI: PlayerUI
player: Player
currentBoss: Boss
GameController
+ GameController(boolean):
- ipAddress: String
- currentGameState: GameState
- itemController: ItemController
- screenController: ScreenController
- entityController: EntityController
- highscore: Highscore
- playerId: String
- initOrUpdateGame(): void
+ loadNewMap(): void
+ resetGame(): void
+ showFPS_UPS(): void
+ callRepaint(): void
+ restartLevelAfterDeath(): void
+ update(): void
+ loadAndInitializeControls(): void
+ useMultiplayer(): void
+ loadControlsFromJson(): List<Map<String, Object>>
+ switchSound(): void
+ saveControlsToJson(String): void
+ startGame(): void
+ newGame(): void
entityController: EntityController
screenController: ScreenController
playerId: String
highscore: Highscore
ipAddress: String
isDrawingLeftOfCurrentPlayersForInterfaceGame: boolean
isPlayingMultiplayer: boolean
currentGameState: GameState
itemController: ItemController
MapController
+ MapController(EntityController, Highscore):
- entityController: EntityController
- mapUI: MapUI
- currentMapIndex: int
- maps: ArrayList<Map>
- mapOffsetX: int
- mapOffsetY: int
+ checkCloseToBorder(): void
+ loadCurrentMapIndex(int): void
+ draw(Graphics, boolean): void
currentItemSpawns: ArrayList<Point>
entityController: EntityController
currentMapIndex: int
currentLevelNumber: int
currentPlayerSpawn: Point
maps: ArrayList<Map>
currentMap: Map
currentFinishSpawn: Point
mapOffsetX: int
mapOffsetY: int
mapUI: MapUI
currentEnemySpawns: ArrayList<Point>
currentBossSpawn: Point
ScreenController
+ ScreenController(ItemController):
- highScoreScreen: HighScoreScreen
- loadingScreen: LoadingScreen
- deathScreen: DeathScreen
- startScreen: StartScreen
- controlScreen: ControlScreen
- interfaceGame: InterfaceGame
+ draw(Graphics, GameState, int, int, Highscore, int, String, int, boolean): void
+ update(Highscore, Player, Item[], ArrayList<ServerObject>): void
highScoreScreen: HighScoreScreen
loadingScreen: LoadingScreen
mapCountHighScoreScreen: int
controlScreen: ControlScreen
interfaceGame: InterfaceGame
deathScreen: DeathScreen
startScreen: StartScreen
mapCountEndScreen: int
GameObjectController
+ GameObjectController(MapController, boolean):
- finish: Finish
+ checkIfPlayersInFinish(Player): boolean
+ updatePoints(MapController, boolean): void
+ drawObjects(Graphics, int, int): void
finish: Finish
SoundController
+ SoundController():
- currentGameState: GameState
- isPlayScreenPlaying: boolean
- isSoundEffectPlaying: boolean
- isStartScreenPlaying: boolean
+ playBackgroundMusic(String): void
+ playSoundEffect(String, int): void
- loadSoundEffect(String): void
+ stopBackgroundMusic(): void
- preloadSoundEffects(): void
+ soundControl(): void
isPlayScreenPlaying: boolean
currentGameState: GameState
isStartScreenPlaying: boolean
isSoundEffectPlaying: boolean
ClientHandler
+ ClientHandler(Socket, Host):
+ run(): void
Host
+ Host():
- serverObjects: ArrayList<ServerObject>
+ addOrUpdateServerObject(ServerObject): void
+ run(): void
+ refreshServerObjects(): void
serverObjects: ArrayList<ServerObject>
LOGIC
<<enumeration>>
ObjectAnimations
- ObjectAnimations():
+ values(): ObjectAnimations[]
+ valueOf(String): ObjectAnimations
aniSize: int
aniIndex: int
Entity
+ Entity(float, float, Float, boolean, int):
+ Entity():
# x: float
# y: float
# hitbox: Float
# health: int
# isDead: boolean
+ decreaseHealth(int): void
isDead: boolean
x: float
health: int
y: float
hitbox: Float
Finish
+ Finish(float, float, boolean):
+ checkIfPlayerInFinish(Player): boolean
+ updateFinishPoint(int, int, boolean): void
isActive: boolean
Boss
+ Boss(float, float):
- name: String
- minProjectileHitboxes: ArrayList<Float>
- projectileHitbox: Float
- isScoreIncreased: boolean
- MAX_HEALTH: int
- minProjectileAttack(int, Player): void
+ resetAllMinProjectiles(): void
+ update(int, Player, Highscore, Finish): void
- moveProjectile(): void
- attack(int, Player): void
- moveMinProjectiles(): void
- setBossPosition(float, float): void
- checkIfProjectileIsOutOfBounds(int): void
- resetNewMinProjectiles(): void
- checkIfAllMinProjectilesAreOutOfBounds(int): void
+ resetProjectile(): void
- bigProjectileAttack(int, Player): void
name: String
isDead: boolean
minProjectileHitboxes: ArrayList<Float>
MAX_HEALTH: int
isUsingBigProjectile: boolean
isScoreIncreased: boolean
projectileHitbox: Float
Enemy
+ Enemy(float, float, float):
- maxHealth: int
- isScoreIncreased: boolean
- isEntityNear: boolean
- attackHitbox: Float
- hasAttacked: boolean
- moveRight: boolean
+ resetHealth(): void
+ checkForCollisionOnPosition(Map, float, float): boolean
+ update(Map, Player, Highscore): void
- checkIfEnemyCollidesOnRight(Map, float, float, float, float): boolean
- updateAttackHitbox(): void
+ hasAttacked(): boolean
- isEntityHitboxNextToEnemy(Entity): boolean
- checkIfEnemyCollidesUnderHim(Map, float, float, float, float): boolean
+ isEntityInrideChecker(Entity): boolean
- updateXPos(Map, float): void
- checkIfEnemyCollidesOnLeft(Map, float, float, float, float): boolean
+ isAttacking(): boolean
- checkIfEnemyIsOverEdge(Map, float, float, float, float, boolean): boolean
- handleAttack(Player): void
isEntityNear: boolean
isAttacking: boolean
y: float
attackHitbox: Float
hasAttacked: boolean
maxHealth: int
x: float
moveRight: boolean
isScoreIncreased: boolean
Map
+ Map(BufferedImage):
- mapData: int[][]
- enemySpawns: ArrayList<Point>
- finishSpawn: Point
- playerSpawn: Point
- itemSpawns: ArrayList<Point>
- tileSize: int
- bossSpawn: Point
- loadObjects(int, int, int): void
+ getSpriteIndex(int, int): int
- calculateMapOffsets(): void
+ getmaxMapOffsetY(): int
- loadEntities(int, int, int): void
- loadMap(): void
- loadMapData(int, int, int): void
+ getmaxMapOffsetX(): int
width: int
bossSpawn: Point
finishSpawn: Point
mapData: int[][]
playerSpawn: Point
height: int
itemSpawns: ArrayList<Point>
tileSize: int
enemySpawns: ArrayList<Point>
<<enumeration>>
Item
+ Item(float, float, Float, Item Animations):
# isActive: boolean
# y: float
# x: float
# animationType: ItemAnimations
# hitbox: Float
- handleItem(Player): boolean
+ getIsActive(): boolean
x: float
y: float
isActive: boolean
hitbox: Float
animationType: ItemAnimations
<<enumeration>>
ItemAnimations
- ItemAnimations():
+ valueOf(String): ItemAnimations
+ values(): ItemAnimations[]
aniSize: int
aniIndex: int
<<enumeration>>
PlayerAnimations
- PlayerAnimations():
+ valueOf(String): PlayerAnimations
+ values(): PlayerAnimations[]
aniSpeed: int
aniSize: int
aniIndex: int
GameObject
+ GameObject(float, float, Float, ObjectAnimations):
# y: float
# hitbox: Float
# x: float
# objectAnimation: ObjectAnimations
x: float
objectAnimation: ObjectAnimations
y: float
hitbox: Float
Player
+ Player(float, float):
- isAir: boolean
- right: boolean
- attack: boolean
- leftBtyEnemy: boolean
- rightAttackHitBox: Float
- attackHitBoxIsActive: boolean
- hasAttacked: boolean
- currentDamagePerAttack: int
- left: boolean
- totalHealth: int
- leftAttackHitBox: Float
- airMovement: float
- deathAnimationFinished: boolean
- adjustPlayerHitboxPosition(Map, Boss, ArrayList<Enemy>): void
- checkIfPlayerCanMoveToPosition(Map, Boss, ArrayList<Enemy>, float, float, float, float): boolean
+ decreaseHealth(int): void
- checkIfPlayerCollidesOverHim(Map, Boss, ArrayList<Enemy>, float, float, float): boolean
+ resetMaximumDamagePerAttack(): void
- checkIfPlayerCollidesUnderHim(Map, Boss, ArrayList<Enemy>, float, float, float, float): boolean
- updateXPos(Map, Boss, ArrayList<Enemy>, float): void
- updateGroundMovement(): void
+ updateSpawnPoint(int, int): void
+ landed(): void
- checkIfPlayerCollidesOnRight(Map, Boss, ArrayList<Enemy>, float, float, float, float): boolean
+ collidesOnPosition(Map, Boss, ArrayList<Enemy>, float, float): boolean
+ update(Map, Boss, ArrayList<Enemy>, Highscore): void
- checkIfPlayerCollidesOnLeft(Map, Boss, ArrayList<Enemy>, float, float, float, float): boolean
- calculateNewPosition(Entity): float
- updateYPos(Map, Boss, ArrayList<Enemy>, float): void
+ resetDeath(): void
- handlePlayerAttackEntity(Entity): void
+ jump(): void
+ resetHealth(): int
+ attack(): void
+ increaseMaximumDamagePerAttack(int): void
attack: boolean
jumping: boolean
left: boolean
playerHealth: int
airMovement: float
inAir: boolean
currentDamagePerAttack: int
leftBtyEnemy: boolean
attackHitBoxIsActive: boolean
isRestingFlootsAir: boolean
isDwelling: boolean
totalHealth: int
y: float
deathAnimationFinished: boolean
hasAttacked: boolean
leftAttackHitBox: Float
right: boolean
x: float
isResting: boolean
rightAttackHitBox: Float
<<enumeration>>
EnemyAnimations
- EnemyAnimations():
+ values(): EnemyAnimations[]
+ getAniSize(EnemyEnum): int
+ valueOf(String): EnemyAnimations
aniIndex: int
Highscore
+ Highscore():
- bestHighscores: ArrayList<Integer>
- currentHighscore: int
- deathCounter: int
+ decreaseHighscoreForDeath(): void
+ increaseDeathCounter(): void
+ increaseHighscoreForBoss(): void
+ addCurrentHighscoreToList(): void
+ findBestHighscores(): void
+ resetHighscore(): void
+ writeHighscore(): void
+ increaseHighscoreForItems(Item[]): void
+ writeAllHighscores(): void
+ decreaseHighscoreForHeart(): void
+ deleteHighscoreFile(): void
+ readHighscore(): Highscore
+ decreaseHighScoreAfterOneSecond(): void
+ increaseHighscoreForEnemies(): void
deathCounter: int
bestHighscores: ArrayList<Integer>
currentHighscore: int
allHighscores: ArrayList<Integer>
SpriteData
+ SpriteData(String, int, int, int, int, int, float, float, float):
- deadAniSize: int
- path: String
- runAniSize: int
- idleAniSize: int
- attackAniSize: int
- scale: float
runAniSize: int
attackAniSize: int
scale: float
XDimension: int
YAdjustmentLiving: float
YAdjustmentDead: float
deadAniSize: int
idleAniSize: int
path: String
PlayerSerializer
+ PlayerSerializer():
+ writePlayer(Player): void
+ readPlayerHealth(): int
<<enumeration>>
GameState
- GameState():
+ values(): GameState[]
+ valueOf(String): GameState
ServerObject
+ ServerObject(Highscore, Player, String):
- currentLevel: int
- deathCounter: int
- playerId: String
- highScore: int
- horizontalPlayerPosition: float
+ updateObject(ServerObject): void
playerId: String
deathCounter: int
currentLevel: int
horizontalPlayerPosition: float
highScore: int
Heart
+ Heart(float, float):
+ handleItem(Player): boolean
PowerRing
+ PowerRing(float, float):
+ handleItem(Player): boolean
<<enumeration>>
GameControls
- GameControls(String, int, boolean):
- isChangeable: boolean
- keyCode: int
+ getControlByCode(int): GameControls
+ valueOf(String): GameControls
+ values(): GameControls[]
+ isKeyAllowed(int): boolean
keyCode: int
isChangeable: boolean
SharedData
+ SharedData():
GameEngine
+ GameEngine(GameController):
- frames: int
- updates: int
+ startGameLoop(): void
+ run(): void
frames: int
updates: int
Client
+ Client(String, GameController):
+ sendDataToServer(ServerObject): ArrayList<ServerObject>
+ run(): void
Control
+ Control(String, int):
- keyCode: int
- name: String
GitHubClient
+ GitHubClient():
- fetchSha(): String
+ readFile(): String
+ writeFile(String): void
HostChecker
+ HostChecker():
KeyboardInputsIngame
+ KeyboardInputsIngame(GameView, GameController):
+ keyPressed(KeyEvent): void
+ handleUserInputKeyPressed(KeyEvent): void
+ handleUserInputKeyReleased(KeyEvent): void
+ keyTyped(KeyEvent): void
+ keyReleased(KeyEvent): void
+ changeControlKeyGameControls, int): void
- initializeControlActionsOnPress(EntityController, ItemController, GameController, ScreenController): void
- initializeControlActionsOnRelease(EntityController, GameController): void